Chapter 17: Code Management using CVS

This chapter introduces the recommended code management solution for UNIX, **CVS** (Concurrent Versions System). Here we provide basic information only; for more detailed information, see `http://www.fnal.gov/docs/products/cvs/`.

17.1 About CVS

CVS uses **RCS** (Revision Control System) as the underlying protocol and assumes availability of **RCS** commands. **RCS** provides a version control system with which you can record the history of your source files. An **RCS** file contains multiple revisions of text, an access list, a change log, descriptive text, and some control attributes. Only the differences between versions are kept.

The **CVS** product allows many programmers to work on the same code simultaneously, each in his or her own directory, and it merges the changes when they are finished. There is no built-in mechanism to prevent concurrent development.

Users can create tags for release versions of software, and easily extract either a release verision or the latest version.

CVS stores all files in a central repository, a directory populated with a hierarchy of files and directories. The files are organized in modules, where a module is made up of one or more files, and can include files from several directories. It is typical to define one module per project. Although the structure of the repository and modules file interact with your build system (e.g., Makefiles), they are essentially independent.

17.2 Accessing CVS

To set up the **CVS** product:

```
% setup cvs
```

The **CVS** document (in postscript format) can be found in the directory pointed to by `$CVS_DIR/doc`. You can also find documentation on the Web for **CVS**, as mentioned above.

Normally you never access files in a repository directly; you use the **CVS** commands to get your own copy of files.

Note that **CVS** points to the editor defined in your *EDITOR* variable for entering log messages (see the **`cvs commit`** command below). If it is not set, the editor defaults to **vi**.

17.3 Basic CVS Commands

The common commands and their functions are listed below (we do not provide information on options here; see the document referenced above):

`% cvs import` installs new release of source in **CVS** repository the first time

`% cvs co <module>` creates new directory called **`<module>`**, populates it with source files; this allows you to "checkout" your own working copy of the source **`<module>`**

`% cvs checkout <module>` equivalent to **`cvs co <module>`**

`% cvs commit` commits changes you have made; opens editing session for entry of log message

`% cvs rtag <release_number module>`

tags a release

`% cvs export -r <release_number module>`

extracts the specified release without **CVS** administrative directories